60/176,625
01/19/00

U.S. PATENT APPLICATION

Title: **PERFORMANCE PATH METHOD AND APPARATUS FOR EXCHANGING DATA AMONG SYSTEMS USING DIFFERENT DATA FORMATS**

Inventors: Robert G. Schaefer
Harsh Wardham
Reynolds and Reynolds Holdings, Inc.
115 South Ludlow Street
Dayton, OH 45402
Attn: Karen Cramer

Assignee: Reynolds and Reynolds Holdings, Inc.
115 South Ludlow Street
Dayton, OH 45402
Attn: Karen Cramer

Attorney: STEPHEN C. GLAZIER
Reg. No. 31,361
Pillsbury Madison & Sutro LLP
1100 New York Avenue, N.W.
Ninth Floor, East Tower
Washington, D.C. 20005
Telephone: (202) 861-3056
Facsimile: (202) 822-0944
glazier_sc@pillsburylaw.com

PERFORMANCE PATH METHOD AND APPARATUS FOR EXCHANGING DATA AMONG SYSTEMS USING DIFFERENT DATA FORMATS

Field of the Invention

This invention relates to data distribution. Specifically, this invention relates to data distribution among computer systems that use different formats to store data.

Background of the Invention

Compatibility of data has eluded organizations whose computer systems need to exchange information with other organizations' computer systems. Many organizations maintain data within their own computer systems in formats and in structures that are unique to their own businesses. As a result, when these computer systems interact with other organizations' computer systems, problems arise due to incompatibility.

Incompatibility may be caused by using incompatible data elements. For example, one system may define a particular data element as alphanumeric, i.e., capable of representing alphabetic or numeric values, while a second system may define the same data element as numeric only. Thus, if the first system transmits the data element with an alphabetic value to the second system, the second system will reject the data element because the data element is not numeric. As a result, the systems are unable to communicate with each other and become incompatible because of incompatible data elements.

In addition to incompatible data elements, the format of files in a computer system may also lead to incompatibility. More computer systems are being developed using

newer technology, such as relational databases. On the other hand, many mainframe-based computer systems built in the past, known as legacy systems, use flat files or multi-value files (a type of flat file) rather than relational databases. This disparity in file format leads to incompatibility problems. For example, one file in a computer system may be stored as a relational database, while similar or necessary information may be stored as a flat file in a second computer system. If the first computer system attempts to access the flat file in the second computer system using commands designed for relational databases, such as structured query language (SQL) commands, the second system will reject the attempt because the flat file cannot be accessed using relational commands. Thus, the two systems are unable to communicate with each other because of incompatible file formats.

Another problem that arises when computer systems exchange information with each other is the possibility of breaches in the security of the computer systems. When a computer system can be accessed by external entities, such as other computer systems, it is vulnerable to security breaches, including data corruption and infection by computer viruses. Thus, there is a need to be able to exchange information among computer systems while maintaining the security of the computer systems.

As the number of entities exchanging information increases, so does the number of different computer systems using different data formats and different file formats. Accordingly, one computer system may be unable to communicate with another computer system due to incompatible data and/or file formats. Thus, there is a need for compatibility among different computers systems, while ensuring the security of the computer systems.

More specifically, client applications have been developed that utilize relational database formats to organize information. These systems use SQL commands to request information from relational databases. While the SQL request extracts information from relational databases efficiently and quickly, there is a problem with this approach. Specifically, the new SQL client applications with relational databases cannot access information from legacy systems with flat or multi-value databases. This, in turn, results in the need for the SQL client applications or the flat databases, or both, to be extensively modified to allow communication between the SQL client applications and the flat databases. Hence, a frustrated need exists to allow new SQL relational database client applications to access older applications with incompatible flat or multi-value databases, with no modification of the SQL application or the flat database.

SUMMARY OF THE INVENTION

The present invention, for the first time, permits SQL relational client applications 101 to access incompatible flat or multi-value databases 117 using ODBC or JDBC clients 107, 109 and ODBC servers 113. This, for the first time, provides virtual compatibility between SQL relational client applications 101 and flat or multi-value databases 117.

The present invention provides data access components 105 that are application program interfaces (API) that allow computer-to-computer communication, and further allow data requests and data to pass between a SQL relational client application 101 and an ODBC client 107 or JDBC client 109. Each API is a layer of computer programs (or "code") developed specifically for each client application 101, and requires no

modification of the data format of the client application 101. Furthermore, the present invention provides data views 115 (that is mapping and stored procedures) to pass data requests and data between an ODBC server 113 and a subject flat or multi-value database 117. Each set of data views 115 is developed specifically for each subject database 117 and requires no modification of the data format of the subject database 117, for the data views 115 to be able to receive SQL data requests and transmit relational data responses, thus providing virtual SQL relational compatibility for each flat or multi-value subject database 117. Hence, as shown in Fig. 1, the API 105 and the data views 115 together allow the SQL relational client application 101 to indirectly access the subject flat or multi-value database 117, using the ODBC client 107, JDBC client 109, and the ODBC server 113.

An additional client application 101 can be brought into the system and access the subject database 117, by developing only a new API 105 for that additional client application 101, and no new data views 115 are needed for that additional client application 101 to access a database 117 that already has its data views 115. Also, an additional database 117 can be brought into the system by developing only a new set of data views 115 for that database 117, and no new API 105 is needed for that additional database 117 to be accessed by a client application 101 that already has its API 105.

It is a further object of the invention to improve the security of computer systems 117 that exchange information with external computer systems 101. External access to data in the computer systems 117 is controlled by using API's 105. Only authorized client applications 101 will have access through API 105 to the computer systems of databases 117. As a result, information is exchanged in real time among computer

systems 101, 117 without introducing the risk of data corruption within the computer systems 117 due to unauthorized access.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a preferred embodiment of the present invention.

FIGS. 2 and 3 are flow diagrams showing a method according to the present invention.

FIG. 4 shows examples of data views according to the present invention.

DETAILED DESCRIPTION OF THE DRAWINGS

In a preferred embodiment of the invention depicted in Fig. 1, a client application domain 102 comprises a client application 101 communicating electronically with the data access components (API) 105 developed by The Reynolds and Reynolds Company, of Dayton, Ohio ("Reynolds") specifically for that client application 101. For example, the client application 101 may be an RGIS inventory product used to control physical inventory. In addition, the client application 101 may be a WD Net parts locator product on the Internet, a credit software application such as Profit Lease, the Ford Motor Company ("Ford") Focal Point car location application, or the Ford FICO service schedule application. These applications have relational databases and use SQL commands to access data. The client application 101 may also utilize recently introduced technologies, such as the standard Hypertext Markup Language (HTML) or programming languages such as Java.

The client application 101 could be designed to communicate electronically with the Liberty ODBC/JDBC client 107/109, but such a communication would require ODBC/JDBC software to be installed at the client application 101. An alternative to installing such software at the client application 101 is the use of the Reynolds data access components (API) 105 (described below).

Electronic communication between the client application 101 and the data access components (API) 105 may take place, for example, through the Internet, leased telephone lines, wireless connection, physical exchange of diskettes or other methods.

In one embodiment, various client applications 101, each in a client application domain 102, can all access the data access domain 110. The various client application domains 102 may be scattered in various locations, and access to the data access domain 110 may be made electronically by the Internet, telephone data line, wireless connection, physical exchange of diskettes or other methods. The client application domain 102 may reside on a personal computer or any other computing environment, including a "green screen," which is a computer that provides a basic disk operating system (DOS) environment.

The data access domain 110 may be located remotely from the system domain 119, and the two domains 110, 119 communicate electronically with each other by the Internet, telephone data line, wireless connection, or other methods. In one preferred embodiment, the data access domain 110 resides on the same computer as system domain 119 with the dealer database 117. Alternatively, the data access domain 110 could also reside on a personal computer that contains the client application domain 102. In other embodiments, the data access domain 110 resides at a third location remote from all

client domains 102 and system domains 119, and the access domain 110 acts as a stand-alone service center or operations center.

Multiple system domains 119 can be accessed by the data access domain 110. Hence, the data access domain 110 can act as a central service center between multiple client application domains 102 and multiple system domains 119.

In one embodiment, the system domain 119 resides on the same computer as the ERA product or ERA^2 product, both by the Reynolds and Reynolds Company, at the dealer location.

In a preferred embodiment, the ODBC client 107 is a Liberty ODBC client, the JDBC client 109 is a Liberty JDBC client, and the ODBC server 113 is a Liberty ODBC server. ODBC/JDBC clients and ODBC servers are software applications that allow SQL relational databases to access databases to obtain data. The Liberty ODBC client 107, Liberty JDBC client 109, and Liberty ODBC Server 113 are available from Liberty Integration Software Inc., of Vancouver, British Columbia.

The Reynolds data access components (API) 105 comprise multiple software components and provide interfaces (not shown) that are defined and implemented according to the needs of the client application 101. The Reynolds data access components (API) 105 performs several functions, including managing the number of communication connections that are allowed to and from the client application 101, managing the data request queue of information to and from the client application 101, providing security mechanisms to limit system access to authorized client applications 101 only, and controlling the scope of the access to information 117 that is provided to each authorized user 101. For example, a specific client application 101 may be

authorized to access only certain fields of the system 117. The Reynolds data access components (API) 105 controls the movement of data to and from the client application 101, such that if the client application 101 requests data from an area 117 that the client application has no authority to access, the Reynolds data access components (API) 105 will disallow access to database 117 for that data request.

In most embodiments, use of the Reynolds data access components (API) 105 will require some communication software additions to the client application 101. For example, if a client application 101 requires access to the dealer database 117 from within a third party product, such as Access or Excel by Microsoft Corporation, the client application 101 will not be able to link or query the dealer database 117 directly. Rather, in this example, the client application 101 would use VBScript to create an instance of the Reynolds data access component 105, use the available interfaces 107, 113, and 115 to retrieve data, and program the client application 101 according to the client application's specific needs for processing the data received. Client applications 101 using a web server environment under Microsoft Windows NT likely will use COM or DCOM components 105 (described below) to perform the required data access.

The Reynolds data access components (API) 105 utilize Open DataBase Connectivity (ODBC), which is a database programming interface standard that provides a common language for Windows applications to access databases on a network, or Java DataBase Connectivity (JDBC), which is a programming interface standard that lets Java applications access databases.

In general, Reynolds data access components (API) 105 fall into one of the following categories, depending on the requirements of the client application:

a. Common Object Request Broker Architecture (CORBA) components will provide data access within cross platform environments. CORBA, an industry standard for communicating among distributed objects, provides a way to execute objects that are written in different programming languages, even if these objects run on different platforms. The Reynolds data access components (API) 105, when they are CORBA-compatible, will require the deployment of an Object Request Broker (ORB), and can be used in an application environment as well as with services such as web servers.

b. COM/DCOM Components will provide data access within the Microsoft Windows environment. These components can be used in conjunction with applications that support object creation as well as with services and web based applications.

c. Java classes will provide access within cross platform environments where CORBA components are not feasible. In this case, the appropriate Java archive files (JAR) will be distributed.

In a preferred embodiment, the system domain 119 comprises the Liberty ODBC Server 113, the mapping and stored procedures interface 115 ("data views"), developed by The Reynolds and Reynolds Company, of Dayton, Ohio ("Reynolds") and the dealer database 117. The Liberty ODBC server 113 communicates electronically with the Liberty ODBC client 107 and the Liberty JDBC client 109, as well as the Reynolds data views 115.

The Reynolds data views 115 are a customized software package, specifically developed for the target dealer database 117. In one embodiment, the Reynolds data views 115 reside on the same computer as the dealer database 117, an example of which may utilize the ERA product by The Reynolds and Reynolds Company. The ERA

product organizes information into a multi-value database, which is a type of flat file. The Liberty products including the Liberty ODBC client 107, the Liberty JDBC client 109, and the Liberty ODBC server 113, on the other hand, provide a relational view of the data in a system. The Reynolds data views 115 convert data from multi-value databases or flat files into a relational database format, in response to SQL relational data requests.

Each distinct dealer database 117 in the system of the present invention has its own specific set of data views 115. In one embodiment, the Reynolds data views 115 electronically communicate with the Liberty ODBC server 113, as well as the dealer database 117. The data views 115 receive an SQL relational data request from the ODBC server 113 and build a relational data response from data in the database 117, and transmit the relational data response back to the ODBC server 113. The data views 115 build the response by extracting the necessary data from the required field in the proper flat file in database 117 and mapping it into the required field of the proper relational table in the relational response. In addition, stored procedures are included within the data views 115 to manipulate or process one or more data fields in the dealer database 117 where necessary before mapping the data into the relational response. For example, a stored procedure may concatenate two fields, or perform a calculation upon these two fields, to create a new data value.

The data views 115 are built according to the requirements of the dealer database 117, which electronically communicates with the data views 115. Thus, for example, if the dealer database 117 uses the Reynolds ERA2 product, the data views 115 would be customized to function in conjunction with the ERA2 product. Similarly, if the dealer database 117 uses an Automatic Data Processing, Inc. (ADP) product, or a customized

software package developed by the dealer, the data views 115 are customized to function with the particular features of the dealer database 117.

Using the data views 115, the system processes the request received from the Liberty ODBC server 113 by extracting the necessary information from the dealer database 117, and mapping and processing that data to build a response in a relational structure ("relational response"). As a result, the client application 101 will not directly access the system domain 119, thus eliminating the risk of database corruption in the dealer database 117. Accordingly, virtual compatibility between incompatible systems is achieved, while the security of the computer system 117 is maintained.

Fig. 2 shows a flow diagram of an embodiment of the invention. The client application 101 initiates an SQL relational data request in block 201. The Reynolds data access components (API) 105 receives the data request from the client application 101. The Reynolds data access components (API) 105 evaluates the data request in block 203. The Reynolds data access components (API) 105, for example, determines if the request originated from a valid source 101 to ensure that data access is authorized. In addition, the Reynolds data access components (API) 105 manage the communications connections to access domain 110, and the request queue in API 105, which queue is comprised of all requests received by the Reynolds data access components (API) 105. The Reynolds data access components (API) 105 also control the scope of access to information in database 117 that is provided to the client application 101.

If the request is valid, the Reynolds data access components (API) 105 transmits the data request as shown in block 205 to an ODBC client 107 or a JDBC client 109. For example, ODBC client 107 may be used if the Reynolds data access components (API)

105 operates in a particular computing environment, such as a Windows NT environment, and if the request from the client application 101 is valid, the Reynolds data access components (API) 105 transmits the request to the ODBC client 107. Similarly, a JDBC client 109 may be used if the Reynolds data access component operates in a JAVA environment, and if the request from the client application 101 is valid, the Reynolds data access components (API) 105 transmits the request to the JDBC client 107.

Further, in block 206, the ODBC client 107 or the JDBC client 109 transmits the data request to the ODBC server 113.

The ODBC server 113 transmits the request to the data views 115 in block 207.

The data views 115 receive the request in block 209. The data views 115 build a relational response by extracting the necessary information from the database 117, and performing any needed stored procedures on the extracted data, and then mapping the extracted and processed data into a relational response.

Thus, even though the SQL relational client application 101 is incompatible with the flat or multi-value database 117, virtual compatibility between systems 101 and 117 is achieved without modification of database 117 or of SQL relational client application 101.

The flow of data from the database 117, shown in Fig. 3, follows the reverse of the path just described. In block 303, the data views 115 transmit the relational data response to the ODBC server 113.

The ODBC server 113 transmits the data to the ODBC client 107 or the JDBC client 109, as shown in block 305.

In block 307, the ODBC client 107 or the JDBC client 109 transmits the data to the API 105.

Finally, in block 309, the Reynolds data access components (API) 105 validate security and exercise control over the flow of data, and if the data is valid and the client application 101 is proper for that data, then the API 105 transmits the data to the client application 101.

Examples of data views 115 specifically for the ERA database 117 are shown in FIG. 4. While a number of data views 115 are shown in FIG. 4, it should be noted that hundreds of data views 115 are used in a set to achieve the goals of the present invention for each specific dealer database 117. The data requirements 405 provide a generic label for the data field requested by a generic SQL data request. The ERA file name 410 identifies in which ERA file the data field resides on the ERA system 117. The field name 415 provides the actual name of the ERA data field in the code used in the ERA product. The field number 420 refers to an internal reference number used by the ERA product to indicate the position of the data field within a multi-value database. (A multi-value database is one in which more than one value can be stored in one field of the database, and it is a type of flat file. It is not a relational database.) The data type 425 describes the nature of the data field, and determines how the data field will be interpreted. The table name 430 identifies the specific relational data table into which the ERA data field will be mapped for the relational data response to the SQL relational data request. The Reynolds and Reynolds Company builds a separate set of data views 115 for each specific dealer database product 117. Thus, if client application 101 requests data in terms of data requirements 405 and table names 430, then the client application 101 does

not need to know how the underlying dealer database 117 is implemented, and the dealer database 117 can be, for example, ERA or ERA^2 from Reynolds, or products from ADP, Oracle or others.

In one preferred embodiment, the data access domain 110 may reside on a computer and the system domain 119 may reside on another computer that is remote from the data access domain 110, and the system domain 119 may electronically communicate with the data access domain 110 by Internet, leased telephone lines, exchange of diskettes or other methods. The client application domain 102 may reside on a computer remote from the data access domain 110 and the system domain 119, and the client application domain 102 may electronically communicate with the data access domain 110 by Internet, leased telephone lines, wireless connection, exchange of diskettes or other methods.

Access authorization by a specific client application 101 through the data access domain 110 may be limited to none, some, or all of the system domains 119. Also, access by a client application 101 to a database 117 may be limited by API 105 to only parts of the database 117.

Which domain 102, 110, 119 specific elements of the present invention 101, 105, 107, 109, 113, 115, 117 may be placed in, can be altered and still remain within the bounds of the present invention. For example, some API's 105 may be located in a client application domain 102.

Other collateral services may be provided at the data access domain 110, such as firewall and security services.

While the present invention has been described in connection with what are presently considered to be the most practical and preferred embodiments, it is to be

understood that the invention is not limited to the disclosed embodiments. On the contrary, the present invention is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims.

WHAT IS CLAIMED IS:

1. An application program interface apparatus comprising:

a. means for receiving a relational data request from a client application;

b. means for managing communications connections and request queues;

c. first means for checking the security authorization and control associated with the data request;

d. means for transmitting the data request if the first checking means determines that the data request is valid;

e. means for receiving a relational data transmission from an ODBC client or a JDBC client;

f. second means for checking the security authorization and control associated with the relational data transmission; and

g. means for transmitting the relational data transmission to the client application if the second checking means determines that the relational data transmission is valid.

2. The apparatus of claim 1 where the client application is selected from the group comprising, automobile manufacturer applications, and applications of vendors to the automobile industry.

3. The apparatus of claim 1, wherein the first and second means for checking security authorization of the data request use identification authentication.

4. The apparatus of claim 1, where the client application has a relational database.

5. The apparatus of claim 1 where the data request and data transmission are transmitted electronically using one means from the group comprising: the Internet, leased telephone lines, wireless communication, and exchange of diskettes.

6. A data view apparatus for mapping and performing stored procedures comprising:

a. means for receiving a relational data request from an ODBC server, such data request originating from a client application;

b. means for extracting data from proper files and fields in a flat or multi-value database, and mapping and performing stored procedures upon the extracted data to build a relational data response; and

c. means for transmitting the relational data response to the ODBC server.

7. The apparatus of claim 6 where the database is an automobile dealer database with flat files.

8. A data interchange system comprising:

a. an application program interface comprising

1) means for receiving a relational data request from a client application;

2) means for managing communications connections and request queues;

3) first means for checking the security authorization and control associated with the data request;

4) means for transmitting the data request to an ODBC client or a JDBC client, if the first checking means determines that the data request is valid;

5) means for receiving a relational data transmission from an ODBC client or a JDBC client;

6) second means for checking the security authorization and control associated with the relational data transmission; and

7). means for transmitting the relational data transmission to the client application if the second checking means determines that the relational data transmission is valid;

b. the JDBC client, electronically communicating with the application program interface and an ODBC server, with means to receive the data request and transmit it to the ODBC server, and means to receive the data response and transmit it to the application program interface;

c. the ODBC client, electronically communicating with the application program interface and the ODBC server, with means to receive the data request and transmit it to the ODBC server, and means to receive the data response and transmit it to the application program interface;

d. the ODBC server, with means to receive the data request and transmit it to the data view apparatus, and means to receive the data response and transmit it to the ODBC client or JDBC client;

e. a data view apparatus for mapping and performing stored procedures comprising:

1) means for receiving the relational data request from the ODBC server;

2) means for extracting data from proper files and fields in a flat or multi-value database, and mapping and performing stored procedures upon the extracted data to build a relational data response; and

3) means for transmitting the relational response to the ODBC server;

f. the database electronically communicating with the data view apparatus.

9. The data interchange system of claim 8 where the client application has a relational database.

10. The data interchange system of claim 8 where the database is an automobile dealer database with flat files.

11. The data interchange system of claim 8 where the client application is selected from the group comprising, automobile manufacturer applications, and applications of vendors to the automobile industry.

12. The data interchange system of claim 8, further including means for checking access authorization of the data request.

13. A data interchange method comprising:

a. an application program interfacing method comprising

1) receiving a data request from a client application;

2) managing communications connections and request queues;

3) checking security authorization and control associated with the data request;

4) transmitting the data request if checking the security authorization and control associated with the data request determines that the data request is valid;

5) receiving a data transmission from an ODBC client or a JDBC client;

6) checking security authorization and control associated with the data transmission; and

7) transmitting the data transmission to the client application if checking the security authorization and control associated with the data transmission determines that the data transmission is valid;

b. electronically communicating among the JDBC client, an application program interface and an ODBC server;

c. electronically communicating among the ODBC client, the application program interface and the ODBC server;

d. electronically communicating between the ODBC server and a data view apparatus;

e. a data viewing method for mapping and performing stored procedures comprising:

1) receiving a data request from an ODBC server, such data request not compatible with a dealer database and such data request originating from a client application;

2) extracting data from proper files and fields in a dealer database, and mapping and performing stored procedures upon the data to build a relational response; and

3) transmitting the relational response to the ODBC server;

f. electronically communicating between a database and the data view apparatus.

14. The data interchange method of claim 13 where the client application has a database of a type selected from the group comprising, flat files, sequential access files, relational databases, multi-value databases, and other file structures.

15. The data interchange method of claim 13 where the dealer database is of a type selected from the group comprising, flat files, sequential access files, relational databases, multi-value databases, and other file structures.

16. The data interchange method of claim 13 where the client application is selected from the group comprising, automobile manufacturer applications, and applications of vendors to the automobile industry.

17. The data interchange method of claim 13, further including checking access authorization of the data request.

18. A data storage medium containing instructions programmed to perform a data interchange method, the method comprising:

a. an application program interfacing method comprising:

1) receiving a data request from a client application;

2) managing communications connections and request queues;

3) checking security authorization and control associated with the data request;

4) transmitting the data request if checking the security authorization and control associated with the data request determines that the data request is valid;

5) receiving a data transmission from an ODBC client or a JDBC client;

6) checking security authorization and control associated with the data transmission; and

7) transmitting the data transmission to the client application if checking the security authorization and control associated with the data transmission determines that the data transmission is valid;

b. electronically communicating among the JDBC client, an application program interface and an ODBC server;

c. electronically communicating among the ODBC client, the application program interface and the ODBC server;

d. electronically communicating between the ODBC server and a data view apparatus;

e. a data viewing method for mapping and performing stored procedures comprising:

1) receiving a data request from an ODBC server, such data request not compatible with a dealer database and such data request originating from a client application;

2) extracting data from proper files and fields in a dealer database, and mapping and performing stored procedures upon the data to build a relational response; and

3) transmitting the relational response to the ODBC server;

f. electronically communicating between a database and the data view apparatus.

19. The data storage medium of claim 18 where the client application has a database of a type selected from the group comprising, flat files, sequential access files, relational databases, multi-value databases, and other file structures.

20. The data storage medium of claim 18 where the dealer database is of a type selected from the group comprising, flat files, sequential access files, relational databases, multi-value databases, and other file structures.

21. The data storage medium of claim 18 where the client application is selected from the group comprising, automobile manufacturer applications, and applications of vendors to the automobile industry.

22. The data storage medium of claim 18, further including means for checking access authorization of the data request.

ABSTRACT

Data access components allow data requests and data to pass between a SQL relational client software application and an ODBC client or JDBC client. Each data access component is developed specifically for each client application. Furthermore, data views (that is, sets of mapping and stored procedures) receive SQL data requests and allow data to pass to an ODBC server from a subject flat database that is not compatible with the relational client application. Each set of data views is developed specifically for each subject flat database, and provides virtual relational compatibility for each subject flat database. The data access components and the data views together allow the relational client application to access the incompatible flat database through the ODBC client, JDBC client, and the ODBC server.

P:\33\258033\performance path specs.doc